SUSE Linux Enterprise Server Libica Cryptographic Module

FIPS 140-2 Non-Proprietary Security Policy

Doc version 1.0.3

Last update: 2021-09-15

Prepared by:
atsec information security corporation
9130 Jollyville Road, Suite 260
Austin, TX 78759
www.atsec.com

Table of contents

1 Cryptographic Module Specification

This document is the non-proprietary security policy for the SUSE Linux Enterprise Server Libica Cryptographic Module version 1.0. It contains the security rules under which the module must operate and describes how this module meets the requirements as specified in FIPS 140-2 (Federal Information Processing Standards Publication 140-2) for a security level 1 module.

FIPS 140-2 details the requirements of the Governments of the U.S. and Canada for cryptographic modules, aimed at the objective of protecting sensitive but unclassified information. For more information on the FIPS 140-2 standard and validation program please refer to the NIST website at http://csrc.nist.gov/.

Throughout the document, "the Libica module" and "the module" are also used to refer to the SUSE Linux Enterprise Server Libica Cryptographic Module version 1.0.

1.1 Module Overview

The SUSE Linux Enterprise Server Libica Cryptographic Module is a software-hybrid module that provides general purpose cryptographic algorithms to applications running in the user space of the underlying operating system through a C language application program interface (API).

The module is composed by a software library, which provides the API and a subset of the cryptographic algorithms, and the Central Processor Assist for Cryptographic Functions (CPACF), which is part of the IBM z15 processor and provides cryptographic algorithms implemented in firmware and hardware. In addition, the module uses the SUSE Linux Enterprise Server OpenSSL Cryptographic Module version 4.1 as a bound module (also referred to as "the bound OpenSSL module"), which provides additional algorithms not implemented in Libica. The SUSE Linux Enterprise Server OpenSSL Cryptographic Module version 4.1 is a FIPS validated module (certificate #3991).

For the purpose of the FIPS 140-2 validation, the module is a software-hybrid, multi-chip standalone cryptographic module validated at overall security level 1. Table 1 shows the security level claimed for each of the eleven sections that comprise the FIPS 140-2 standard:

FIPS 140-2 Section		Security Level
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1
3	Roles, Services and Authentication	1
4	Finite State Model	1
5	Physical Security	1
6	Operational Environment	1
7	Cryptographic Key Management	1
8	EMI/EMC	1
9	Self Tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A

Table 1: Security Levels

Table 2 lists the components of the cryptographic module that define the logical boundary of the Libica module.

Component Type	Version	Components	Description
Software	1.0	/usr/lib64/libica.so.3.6.0	Shared library for cryptographic algorithms.
		/usr/lib64/.libica.so.3.6.0.hmac	Integrity check HMAC value for the Libica shared library.
Firmware	FC3863	Feature Code 3863 (FC3863), CPACF DES/TDES Enablement	Enables processor accelleration implementation for AES and Triple-DES.
Hardware	IBM z15	Coprocessor that implements the CPACF, integrated into the IBM z15 processor	CPACF is implemented on the IBM z15 processor (SHA-1, SHA-2, and SHA-3 are directly available to application programs).

Table 2: Cryptographic Module Components

The diagram below shows the logical boundary of the module (enclosed in dotted blue lines), and its interfaces with the bound OpenSSL module and the operational environment.

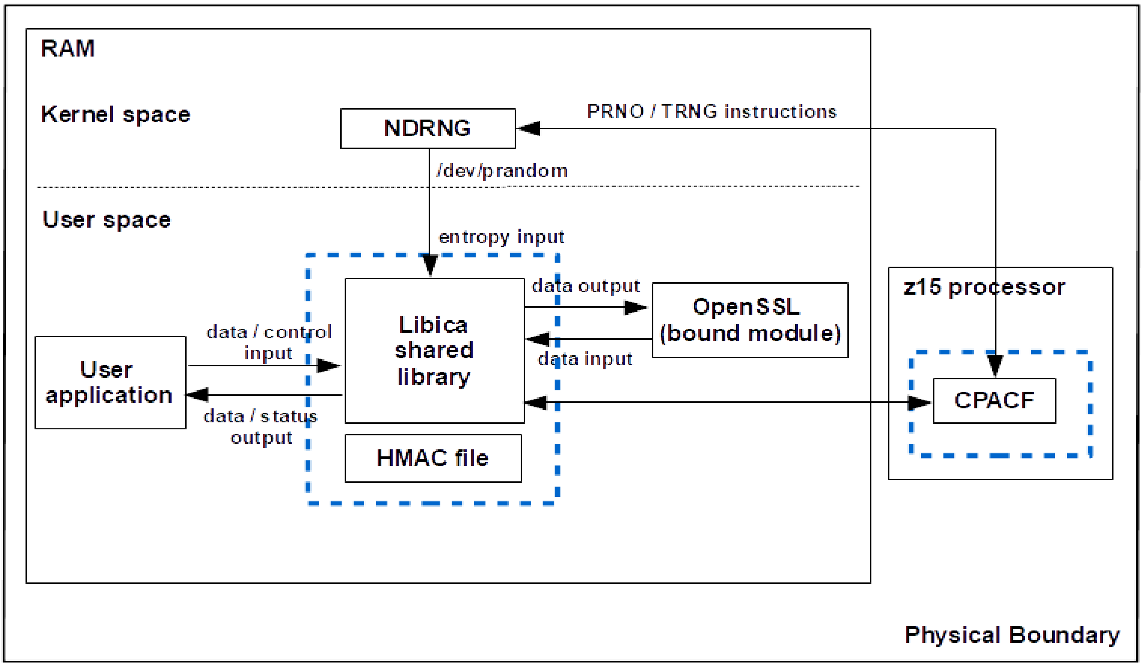

Figure 1: Logical Block Diagram

The module is aimed to run on a general purpose computer (GPC). Table 3 shows the platform on which the module has been tested:

Platform	Processor	Test Configuration
IBM System Z/15 with FC3863	IBM z15	SUSE Linux Enterprise Server 15 SP2

Table 3: Tested Platforms

Note: Per FIPS 140-2 IG G.5, the Cryptographic Module Validation Program (CMVP) makes no statement as to the correct operation of the module or the security strengths of the generated keys when this module is ported and executed in an operational environment not listed on the validation certificate.

The physical boundary of the module is the surface of the case of the tested platform. Figure 2 shows the hardware block diagram including major hardware components of a GPC.

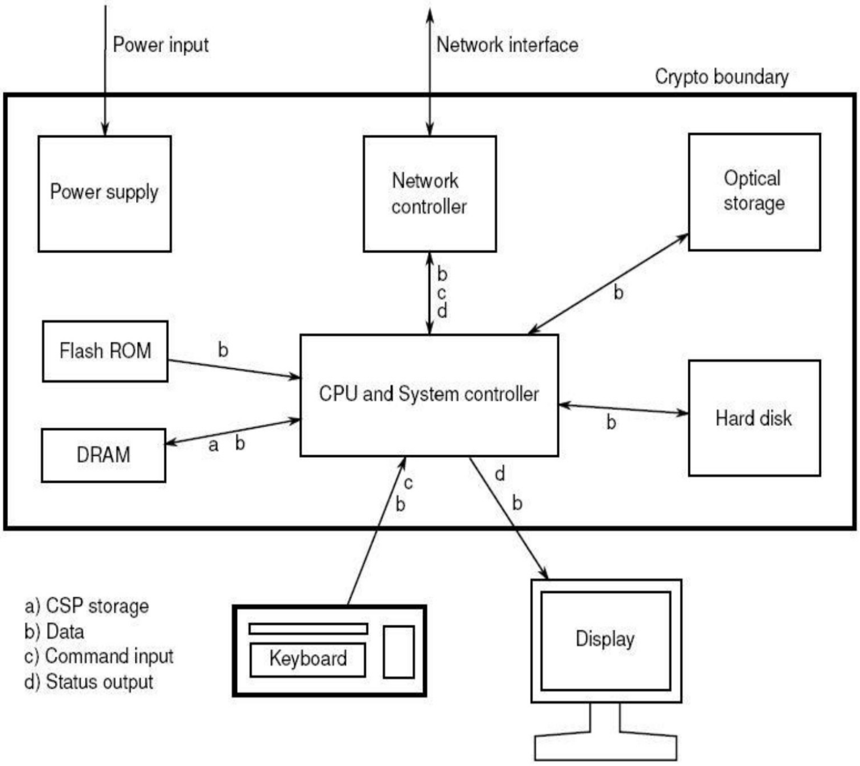

Figure 2: Hardware Block Diagram

1.2 Modes of Operation

The module supports two modes of operation:

- FIPS mode (the Approved mode of operation): only approved or allowed security functions with sufficient security strength can be used.
- non-FIPS mode (the non-Approved mode of operation): only non-approved security functions can be used.

The module enters FIPS mode after power-up tests succeed. Once the module is operational, the mode of operation is implicitly assumed depending on the security function invoked and the security strength of the cryptographic keys.

Critical security parameters (CSPs) used or stored in FIPS mode are not used in non-FIPS mode, and vice versa. The module creates separate contexts for each cryptographic service; therefore, the CSPs used in FIPS approved services and non-approved services are separated by design and not shared.

2 Cryptographic Module Ports and Interfaces

For the purpose of the FIPS 140-2 validation, the physical ports are interpreted to be the physical ports of the hardware platform on which it runs.

The logical interfaces are the API through which applications request services, and the application program interface (API) provided by the bound OpenSSL module. The ports and interfaces are shown in the following table.

FIPS Interface	Physical Port	Logical Interface
Data Input	None	API input parameters for data.
Data Output	None	API output parameters for data.
Control Input	None	API function calls, API input parameters for control input, /proc/sys/crypto/fips_enabled control file, ICAPATH environment variable.
Status Output	None	API return codes, API output parameters for status output.
Power Input	PC Power Supply Port	N/A

Table 4: Ports and Interfaces

3 Roles, Services and Authentication

3.1 Roles

The module supports the following roles:

- User role: performs cryptographic services (in both FIPS mode and non-FIPS mode), key zeroization, get status, and on-demand self-test.
- Crypto Officer role: performs module installation and configuration.

The User and Crypto Officer roles are implicitly assumed by the module based on service requested. No authentication is required.

3.2 Services

The module provides services to the users that assume one of the available roles. All services are shown in Table 5 and Table 6.

Table 5 lists the services available in FIPS mode. For each service, the table lists the associated cryptographic algorithm(s), the role to perform the service, the cryptographic keys or CSPs involved, and their access type(s). The following convention is used to specify access rights to a CSP:

- *Create*: the calling application can create a new CSP.
- *Read*: the calling application can read the CSP.
- *Update*: the calling application can write a new value to the CSP.
- *Zeroize*: the calling application can zeroize the CSP.
- *n/a*: the calling application does not access any CSP or key during its operation.

The details of the approved cryptographic algorithms implemented in the module, including the CAVP certificate numbers, can be found in Table 7. The approved cryptographic algorithms provided by the bound OpenSSL module, with the CAVP certificate numbers, are shown in Table 8.

Service	Algorithms	Role	Keys/CSPs	Access
Cryptographic Services				
Symmetric encryption and decryption	AES	User	AES key	Read
	Three-key Triple-DES	User	Three-key Triple-DES key	Read
RSA key generation	RSA, DRBG	User	RSA public and private keys	Create
ECDSA key generation	ECDSA, DRBG	User	ECDSA public and private keys	Create
ECDSA public key validation	ECDSA	User	ECDSA public key	Read
ECDSA signature generation and verification	ECDSA, DRBG, SHS	User	ECDSA public and private keys	Read
EC Diffie-Hellman shared secret computation	KAS ECC	User	ECDSA public and private keys	Read
			Shared secret	Create
Random number generation	DRBG	User	Entropy input string, seed material	Read
			Internal state	Update

Service	Algorithms	Role	Keys/CSPs	Access
Message digest	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	User	None	N/A
	SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE-128, SHAKE-256	User	None	N/A
Message authentication code (MAC)	CMAC with AES	User	AES key	Read
	CMAC with Triple-DES	User	Triple-DES key	Read
Other FIPS-related Services				
Show status	N/A	User	None	N/A
Zeroization	N/A	User	All CSPs	Zeroize
On-demand self-tests	AES, DRBG, SHS, SHA-3 Triple-DES	User	None	N/A
Module installation and configuration	N/A	Crypto Officer	None	N/A

Table 5: Services in FIPS mode of operation

Table 6 lists the services only available in non-FIPS mode of operation. The details of the non-approved cryptographic algorithms available in non-FIPS mode can be found in Table 10.

Service	Algorithm / Modes	Role	Keys	Access
Cryptographic Services				
Symmetric encryption and decryption	AES in GCM mode, Triple-DES in CBC_CS mode.	User	Symmetric key	Read
Asymmetric key generation	ElGamal	User	Public and private keys	Create
	RSA and ECDSA restrictions listed in Table 10			
	Curves ed25519, x25518, ed448 and x448.			
Digital signature generation and verification	RSA and ECDSA and message digest restrictions listed in Table 10	User	Public and private keys	Read
	Curves ed25519, x25518, ed448 and x448.			
RSA sign, verify, encrypt and decrypt primitives	RSA	User	RSA key pair	Read
EC Diffie-Hellman shared secret computation	EC Diffie-Hellman with P-192 curve, K curves, B curves and non-NIST curves.	User	ECDSA public and private keys	Read
			Shared secret	Create

Table 6: Services in non-FIPS mode of operation

3.3 Operator Authentication

The module does not implement user authentication. The role of User and Crypto Officer are implicitly assumed based on the service requested.

3.4 Algorithms

The module provides C implementation of cryptographic algorithms for SHS and DRBG, and uses algorithm implementations provided by the CPACF, implemented in IBM z15 processors, for AES, SHS, SHA-3 and Triple-DES. The module also uses cryptographic services provided by the OpenSSL bound module for HMAC, ECDSA, KAS ECC, RSA and SHS.

Table 7 lists the approved algorithms, the CAVP certificates, and other associated information of the cryptographic implementations in FIPS mode implemented in the module, using C implementation and CPACF. Please refer to Appendix A for more detailed information about the algorithm implementations tested for each CAVP certificate. Notice that the CAVP certificate A389, issued for the CPACF, may include more algorithms than the ones used by the module.

Algorithm	Mode / Method	Key Lengths, Curves (in bits)	Use	Standard	CAVP Certs
AES	ECB, CBC, CFB8, CFB128, OFB, CTR	128, 192, 256	Data encryption and decryption	FIPS197, SP800-38A	A389
	CBC_CS, CTR	128, 192, 256	Data encryption and decryption	FIPS197, SP800-38A	A806
	CMAC	128, 192, 256	MAC generation and verification	SP800-38B	A389
	CCM	128, 192, 256	Data encryption and decryption	SP800-38C	A806
	XTS	128, 256	Data encryption and decryption for data storage	SP800-38E	A389
DRBG	Hash_DRBG: SHA-512 without PR	N/A	Deterministic random bit generation	SP800-90A	A491
SHA-3	SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE-128, SHAKE-256		Message Digest	FIPS202	A389
SHS	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	N/A	Message digest	FIPS180-4	A389
	SHA-512/224, SHA-512/256	N/A	Message digest	FIPS180-4	A491
Triple-DES	ECB, CBC, CFB8, CFB64, OFB, CTR	192 (three-key Triple-DES)	Data encryption and decryption	SP800-67 SP800-38A	A389
	CMAC	192	MAC generation and verification	SP800-67 SP800-38B	

Table 7: Approved Cryptographic Algorithms provided by the Libica module

Table 8 lists the approved algorithms that are used by the module but provided by the bound OpenSSL module in FIPS mode. Please refer to Appendix A for more detailed information about the algorithm implementations tested for each CAVP certificate.

Algorithm	Mode / Method	Key Lengths, Curves (in bits)	Use	Standard	CAVP Certs
ECDSA		P-256, P-384, P-521	Key pair generation, Public key verification	FIPS186-4	A360
	SHA-224, SHA-256, SHA-384, SHA-512	P-224, P-256, P-384, P-521	Digital signature generation		
	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	P-224, P-256, P-384, P-521	Digital signature verification		
HMAC	SHA-256	112 or greater	MAC generation for integrity tests.	FIPS198-1	A360
KAS-ECC-SSC	ECC Ephemeral Unified scheme	P-224, P-256, P-384, P-521	EC Diffie-Hellman shared secret computation	SP800-56Arev3	A684
RSA	B.3.3	2048, 3072, 4096	Key pair generation	FIPS186-4	A360

Table 8: Approved Cryptographic Algorithms provided by the bound OpenSSL module

3.5 Allowed Algorithms

Table 9 describes the non-approved but allowed algorithms in FIPS mode.

Algorithm	Use
NDRNG	The module obtains entropy data from a NDRNG to seed the DRBG.

Table 9: Non-Approved but Allowed Algorithms

3.6 Non-Approved Algorithms

Table 10 shows the non-Approved cryptographic algorithms implemented in the module or the bound OpenSSL module that are only available in non-FIPS mode.

Algorithm	Use
Triple-DES in CBC_CS mode.	Data encryption and decryption.
AES in GCM mode.	Authenticated data encryption and decryption.
SHA-1.	Message digest in digital signature generation.
RSA with keys smaller than 2048 bits or greater than 4096 bits.	Key pair generation.
RSA modulus exponentiation primitive.	Sign, verify, encrypt, and decrypt primitives.
ECDSA with P-192 and P-224 curves, K curves, B curves and non-NIST curves.	Key pair generation.
ECDSA with P-192 curve, K curves, B curves and non-NIST curves.	Digital signature generation and verification.
EC Diffie-Hellman with P-192 curve, K curves, B curves and non-NIST curves.	Shared secret computation.
All algorithms involving curves ed25519, x25518, ed448 and x448.	Key pair generation, digital signature generation and verification.

Table 10: Non-Approved Cryptographic Algorithms

4 Physical Security

The Libica module inherits the physical characteristics of the host running it; the module has no physical security characteristics of its own. Figure 3 illustrates the IBM System z15 mainframe computer that represents the testing platform, that includes the hardware component of the cryptographic module.

Figure 3: IBM z15 Mainframe Computer

The Central Processor Assist for Cryptographic Functions (CPACF) is part of the CoProcessor Unit (CoP) integrated in the IBM z15 processor, and offers the full implementation of the Triple-DES algorithm, Advanced Encryption Standard (AES) algorithm, Secure Hash Algorithm (SHA) and the PRNO and TRNG instructions used as entropy input. Security Level 1 is satisfied by the device (CoP) being included within the physical boundary of the module and the device being made of production grade components.

With regards to the CPACF physical design, each microprocessor (core) on the 8-core chip (see Figure 4) has its own dedicated CoP, which implements the crypto instructions and also provides the hardware compression function. The compression unit is integrated with the CPACF, benefiting from combining (sharing) the use of buffers and interfaces.

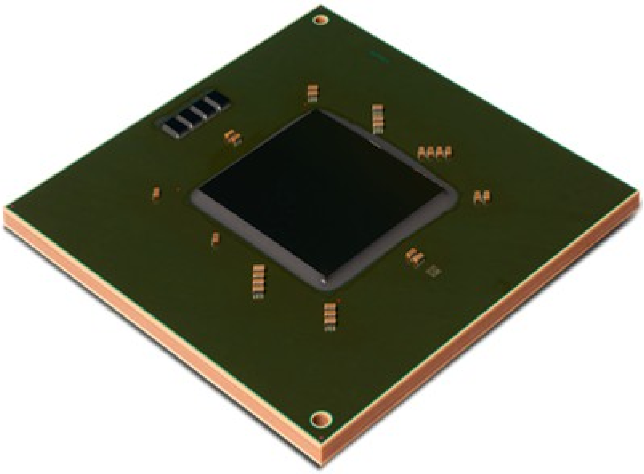

Figure 4: IBM z15 Processor Unit Chip

5 Operational Environment

This module operates in a modifiable operational environment per the FIPS 140-2 level 1 specifications. The module runs on a commercially available general-purpose operating system executing on the hardware specified in Table 3.

5.1 Policy

The operating system is restricted to a single operator; concurrent operators are explicitly excluded.

The application that requests cryptographic services is the single user of the module.

The ptrace system call, the debugger gdb and strace shall not be used.

6 Cryptographic Key Management

Table 11 summarizes the Critical Security Parameters (CSPs) that are used by the cryptographic services implemented in the module. Key sizes allowed in the approved mode of operation are specified in Table 7 and Table 8.

Name	Generation	Entry and Output	Zeroization
AES keys	Not applicable. Key material is entered via API parameters.	Entry via API input parameters in plaintext.	Keys are zeroized when service finishes.
Triple-DES keys			
RSA public and private keys	The module generates keys by using the services provided by the bound OpenSSL module.	Entry via API input parameters in plaintext. Output via API output parameters in plaintext.	Keys are zeroized when service finishes using the bound OpenSSL module.
ECDSA public and private keys			
EC Diffie-Hellman public and private keys			
Shared secret	Generated by the bound OpenSSL module during the EC Diffie-Hellman shared secret computation.	Output via API output parameters in plaintext.	Keys are zeroized when service finishes.
Entropy input string and seed material	Obtained from the NDRNG.	Not applicable, it remains within the logical boundary.	Zeroization occurs when the module terminates.
DRBG internal state: V value, C value	Derived from entropy input as defined in SP800-90A.	Not applicable, it remains within the logical boundary.	

Table 11: Life cycle of Keys or CSPs

The following sections describe how CSPs, in particular cryptographic keys, are managed during its life cycle.

6.1 Random Number Generation

The module employs a Deterministic Random Bit Generator (DRBG) based on [SP800-90A] for the Random Number Generation service provided to calling applications.

The DRBG supports the Hash_DRBG mechanism using SHA-512 without prediction resistance. The module performs the DRBG health tests as defined in section 11.3 of [SP800-90A].

The module uses a Non-Deterministic Random Number Generator (NDRNG) as the entropy source for obtaining 440 bits to seed the DRBG during initialization and reseed during operation. The NDRNG is accessed by the module via the /dev/prandom device of the operational environment. The NDRNG provides at least 256 bits of entropy to seed the DRBG implemented by the module.

The NDRNG is implemented in the underlying Operational Environment (i.e. Linux kernel) and uses the PRNO and TRNG instructions provided by the CPACF in the z15 processor. The NDRNG is within the module's physical boundary but outside of the module's logical boundary.

6.2 Key/CSP Generation

The module does not implement key generation. The module does provide services for the generation of RSA and ECDSA keys, but it uses the services provided by the bound OpenSSL module.

6.3 Key Agreement

The module does not implement key agreement. The module provides a service for Elliptic Curve Diffie-Hellman shared secret computation, but it uses the service provided by the bound OpenSSL module.

6.4 Key Transport

The module does not provide key transport mechanisms.

6.5 Key Derivation

The module does not provide key derivation.

6.6 Key/CSP Entry and Output

The module does not support manual key entry. The keys are provided to the module via API input parameters in plaintext form and output via API output parameters in plaintext form. This is allowed by [FIPS140-2_IG] IG 7.7, according to the “CM Software to/from App Software via GPC INT Path” entry on the Key Establishment Table.

6.7 Key/CSP Storage

Symmetric keys, HMAC keys, public and private keys are provided to the module by the calling application via API input parameters, and are destroyed by the module when invoking the appropriate API function calls.

The module does not perform persistent storage of keys. The keys and CSPs are stored as plaintext in the RAM. The only exception is the HMAC key used for the Integrity Test, which is stored in the module and relies on the operating system for protection.

6.8 Key/CSP Zeroization

The memory occupied by keys is allocated by regular memory allocation operating system calls when a service is invoked. The module zeroizes keys when the service is finished. The zeroization functions overwrite the memory occupied by keys with “zeros” and deallocate the memory with the regular memory deallocation operating system call.

The module also requests the zeroization services provided by the bound OpenSSL module to zeroize CSPs used for RSA, ECDSA and EC Diffie-Hellman related services.

7 Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC)

The test platforms as shown in Table 3 are compliant to 47 CFR FCC Part 15, Subpart B, Class A (Business use).

8 Self Tests

8.1 Power-Up Tests

The module performs power-up tests when the module is loaded into memory, without operator intervention. Power-up tests ensure that the module is not corrupted and that the cryptographic algorithms work as expected.

While the module is executing the power-up tests, services are not available, and input and output are inhibited. The module is not available for use by the calling application until the power-up tests are completed successfully.

If any of the power-up test fails, the module enters the Error state. In that state, data output is inhibited and subsequent calls to the module will also fail; no further cryptographic operations are possible. If the power-up tests complete successfully, the module will enter the Operational state and will accept cryptographic operation service requests.

In order to verify the result of the self-tests, the calling application may invoke the **`ica_fips_status()`** function. The function will return an integer value with the following mask values defined in **`ica_api.h`**:

Define	Value	Description
ICA_FIPS_MODE	0x01	Module has FIPS flag on
ICA_FIPS_CRYPTOALG	0x02	Module failed any of the known answer tests
ICA_FIPS_INTEGRITY	0x04	Module failed the Integrity tests.
ICA_FIPS_RNG	0x10	Module failed the DRBG health tests.

Table 12: Return bit mask values of the ica_fips_status() function

For instance, if the function retuns 0x01, the module was configured properly (FIPS flag on) and the module is operational; if the function returns 0x00, the module did not have any errors during power-up but is not properly configured (FIPS flag off); in any other case, (0x02 or 0x04 bits on) , there were errors during the power-up self-tests.

For the cryptographic algorithms listed in Table 8, the power-up self-tests are performed by the bound OpenSSL module.

8.1.1 Integrity Tests

The integrity of the module is verified by comparing an HMAC-SHA-256 value calculated at run time with the HMAC value stored in the .hmac file that was computed at build time for each software component of the module. If the HMAC values do not match, the test fails and the module enters the error state. The module uses the HMAC-SHA-256 algorithm provided by the OpenSSL bound module.

The integrity of the firmware part of the CPACF is verified using the Cyclic Redundancy Check (CRC-32) algorithm.

8.1.2 Cryptographic Algorithm Tests

The module performs self-tests on all FIPS-Approved cryptographic algorithms supported in the Approved mode of operation, using the Known Answer Tests (KAT) shown in the following table.

Algorithm	Power-Up Tests
AES	KAT AES ECB mode with 128, 192 and 256 bit keys, encryption and decryption (separately tested). KAT AES CBC mode with 128, 192 and 256 bit keys, encryption and decryption (separately tested).

Algorithm	Power-Up Tests
	KAT AES CBC_CS mode with 128, 192 and 256 bit keys, encryption and decryption (separately tested). KAT AES CFB mode with 128, 192 and 256 bit keys, encryption and decryption (separately tested). KAT AES OFB mode with 128, 192 and 256 bit keys, encryption and decryption (separately tested). KAT AES CTR mode with 128, 192 and 256 bit keys, encryption and decryption (separately tested). KAT AES CCM mode with 128, 192 and 256 bit keys, encryption and decryption (separately tested). KAT AES XTS mode with 128 and 256 bit keys, encryption and decryption (separately tested).
CMAC	KAT AES CMAC with 128, 192 and 256 bit keys, MAC generation. KAT Triple-DES CMAC, MAC generation.
DRBG	KAT Hash_DRBG with SHA-512 without PR.
SHA-3	KAT SHA3-224, SHA3-256, SHA3-384 and SHA3-512.
SHS	KAT SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512.
Triple-DES	KAT Triple-DES ECB mode, encryption and decryption (separately tested). KAT Triple-DES CBC mode, encryption and decryption (separately tested). KAT Triple-DES CFB mode, encryption and decryption (separately tested). KAT Triple-DES OFB mode, encryption and decryption (separately tested). KAT Triple-DES CTR mode, encryption and decryption (separately tested).

Table 13: Self-Tests

For the KAT, the module calculates the result and compares it with the known value. If the answer does not match the known answer, the KAT fails and the module enters the Error state.

8.2 On-Demand Self-Tests

On-Demand self-tests can be invoked by invoking the **`ica_fips_powerup_tests()`** function. The function executes the same tests as during power-up.

In order to verify the result of the self-tests, the calling application may invoke the **`ica_fips_status()`** function, as explained in section 8.1.

8.3 Conditional Tests

The module does not implement conditional tests. The module relies on the bound OpenSSL module for the pair-wise consistency tests for assymetric key generation.

8.4 Error states

The Module enters the Error state when the power-on self-tests or the on-demand self-tests fail. In the Error state, all data output is inhibited and no cryptographic operation is allowed.

In order to determine whether the module is in the Error state, the calling application may invoke the **`ica_fips_status()`** function, as explained in section 8.1. The module can be recovered from the Error state by restarting it (i.e. by powering off and powering on). The on-demand self-tests do not recover the module from the Error state.

9 Guidance

9.1 Crypto Officer Guidance

The binaries of the module are contained in the RPM packages for delivery. The Crypto Officer shall follow this Security Policy to configure the operational environment and install the module to be operated as a FIPS 140-2 validated module.

The following RPM packages contain the FIPS validated module:

Processor Architecture	RPM Packages
IBM z15	libica3-3.6.0-5.3.1.s390x.rpm

Table 14: RPM packages

9.1.1 Module Installation

The Crypto Officer can install the RPM packages containing the module as listed in Table 14 using the zypper tool. The integrity of the RPM package is automatically verified during the installation, and the Crypto Officer shall not install the RPM package if there is any integrity error.

9.1.2 Operating Environment Configuration

In order to configure the operating environment, the following bound module must be installed:

- SUSE Linux Enterprise Server OpenSSL Cryptographic Module version 4.1

Please follow the instructions provided in the security policies ([OPENSSL-SP]) to install and configure both modules in FIPS mode of operation.

Once this module is installed and configured properly, the operating environment is configured to support FIPS operation. The Crypto Officer should check the existence of the file/proc/sys/crypto/fips_enabled, which content should be the character “1”. If the file does not exist or does not contain “1”, the operating environment is not configured to support FIPS and the module will not operate as a FIPS validated module properly.

9.1.3 Module Configuration

In the FIPS approved mode of operation, for ECDSA, EC Diffie-Hellman and RSA algorithms, the module shall use the cryptographic services provided by the bound OpenSSL module. The calling application must meet the following requirements:

- the `ica_open_adapter()` and `ica_close_adapter()` functions shall not be used.
- the following environment variable must be set:

```
export ICAPATH=2
```

Any other configuration of the module has not been tested under the current scope of validation. Nevertheless, SUSE provides additional product documentation for using other combinations of the module with other validated hardware cryptographic modules. Please contact SUSE for further information.

9.2 User Guidance

In order to run in FIPS mode, the module must be operated using the FIPS Approved services, with their corresponding FIPS Approved and FIPS allowed cryptographic algorithms provided in this Security Policy (see section 3.2). In addition, key sizes must comply with [SP800-131A].

9.2.1 AES XTS

The AES algorithm in XTS mode can be only used for the cryptographic protection of data on storage devices, as specified in [SP800-38E]. The length of a single data unit encrypted with the XTS-AES shall not exceed 2^{20} AES blocks that is 16MB of data.

To meet the requirement stated in IG A.9, the module implements a check to ensure that the two AES keys used in AES XTS mode are not identical.

9.2.2 Triple-DES encryption

Data encryption using the same three-key Triple-DES key shall not exceed 2^{16} Triple-DES blocks (2GB of data), in accordance to SP800-67 and IG A.13.

[SP800-67] imposes a restriction on the number of 64-bit block encryptions performed under the same three-key Triple-DES key.

When the three-key Triple-DES is generated as part of a recognized IETF protocol, the module is limited to 2^{20} 64-bit data block encryptions. This scenario occurs in the following protocols:

- Transport Layer Security (TLS) versions 1.1 and 1.2, conformant with [RFC5246]
- Secure Shell (SSH) protocol, conformant with [RFC4253]
- Internet Key Exchange (IKE) versions 1 and 2, conformant with [RFC7296]

In any other scenario, the module cannot perform more than 2^{16} 64-bit data block encryptions.

The user is responsible for ensuring the module’s compliance with this requirement.

10 Mitigation of Other Attacks

The module does not implement any mitigation mechanism.

Appendix A - CAVP certificates

The tables below show the certificates obtained from the CAVP for all the target platforms included in Table 3. The CAVP certificates validate all algorithm implementations used as approved or allowed security functions in FIPS mode of operation. The tables include the certificate number, the label used in the CAVP certificate for reference and a description of the algorithm implementation. Notice that not all the algorithms listed in the certificates are used by this module.

Cert#	CAVP Label	Algorithm Implementation
A491	Generic C	Generic C implementation
A389	CPACF	IBM CP Assist Crypto Function (CPACF)
A806	Libica CPACF	Implementation using CPACF for AES in CBC_CS, CCM and CTR modes

Table 15: Libica and CPACF CAVP certificates for the z15 processor

Cert#	CAVP Label	Algorithm Implementation
A360	SHA_ASM	All algorithms impacted by SHA using assembler implementation.
A684	SP800 56A rev 3	SP800-56Arev3 compliant implementation.

Table 16: OpenSSL CAVP certificates for the z15 processor

Appendix B - Glossary and Abbreviations

AES	Advanced Encryption Specification
AES_NI	Intel® Advanced Encryption Standard (AES) New Instructions
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining Message Authentication Code
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CPACF	Central Processor Assist for Cryptographic Functions
CSP	Critical Security Parameter
CTR	Counter Mode
DES	Data Encryption Standard
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
FIPS	Federal Information Processing Standards Publication
GCM	Galois Counter Mode
HMAC	Hash Message Authentication Code
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
PKCS	Public Key Cryptography Standards
RNG	Random Number Generator
RPM	Red hat Package Manager
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
TDES	Triple-DES
XTS	XEX Tweakable Block Cipher with Ciphertext Stealing

Appendix C - References

FIPS 140-2 **FIPS PUB 140-2 - Security Requirements for Cryptographic Modules**
http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

FIPS 140-2_IG **Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program**
December 3, 2019
http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf

FIPS180-4 **Secure Hash Standard (SHS)**
http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

FIPS186-4 **Digital Signature Standard (DSS)**
http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

FIPS197 **Advanced Encryption Standard**
http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

FIPS198-1 **The Keyed Hash Message Authentication Code (HMAC)**
http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf

FIPS202 **SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions**
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf

OPENSSL-SP **SUSE Linux Enterprise Server OpenSSL Cryptographic Module version 4.1 - FIPS 140-2 Non-Proprietary Security Policy**
https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/security-policies/140sp3991.pdf

PKCS#1 **Public Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1**
http://www.ietf.org/rfc/rfc3447.txt

SP800-38A **NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques**
http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf

SP800-38B **NIST Special Publication 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication**
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38b.pdf

SP800-38C **NIST Special Publication 800-38C - Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality**
http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf

SP800-38D **NIST Special Publication 800-38D - Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC**
http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf

SP800-38E **NIST Special Publication 800-38E - Recommendation for Block Cipher Modes of Operation: The XTS AES Mode for Confidentiality on Storage Devices**
http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38e.pdf

SP800-38F **NIST Special Publication 800-38F - Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping**
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf

SP800-56Arev3 **NIST Special Publication 800-56Ar3 - Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography**
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf

SP800-67 **NIST Special Publication 800-67 Revision 1 - Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher**
http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-67r1.pdf

SP800-90A **NIST Special Publication 800-90A Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators**
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

SP800-131A **NIST Special Publication 800-131A Revision 1- Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths**
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

SP800-132 **NIST Special Publication 800-132 - Recommendation for Password-Based Key Derivation - Part 1: Storage Applications**
https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf